Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) An automatic compiler test program generation method, comprising the steps of:

selecting a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions, each of which may form an element of a compiler test program; and

combining the selected sub-procedural descriptions in accordance with the selected order to generate a compiler test program,

wherein the sub-procedural description includes as a type of basic sub-procedural description that lists executable statements in the order of execution, a control-sub-procedural description that describes how to repeat a procedure, a control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function; and

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result.

2. (Original) An automatic compiler test program generation method according to claim 1, wherein each sub-procedural description is assigned a value

which indicates the probability of the sub-procedural description being selected and the value is reflected in the random number-used selection of sub-procedural descriptions.

3. (Original) An automatic compiler test program generation method according to claim 1, wherein variables included in the selected plural sub-procedural descriptions are made shared by the selected sub-procedural descriptions when the selected sub-procedural descriptions are combined.

4. (Cancelled)

5. (Currently Amended) An automatic compiler test program generation method, comprising the steps of:

selecting a plurality of program cells by a random number-used method from a set of program cells, each of which may form an element of a test program for a compiler after being compiled by the compiler; and

combining the selected program cells in accordance with the selected order to generate a test program;

wherein the program cell includes as a type of a basic cell that lists executable statements in the order of execution, a control cell that describes how to repeat a procedure, a control cell that describes how to select a procedure, and a function cell that describes statements to call a function; and

wherein the program cell includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result.

~~wherein each program cell has a description which specifying whether the program cell is to be compiled alone or in combination with the other selected program cells.~~

6. (Cancelled)

7. (Currently Amended) An automatic compiler test program generation method according to claim ~~6~~5, wherein the selected program cells are registered with a table in the order of selection and the registered cells are given the following combining process in the order of registration:

(1) if the cell preceding the current program cell is said basic cell or said function cell, the current cell is concatenated to the preceding cell; and

(2) if the cell preceding the current program cell is said control cell, the current program cell is nested into the preceding cell.

8. (Currently Amended) An automatic compiler test program generation method according to claim ~~7~~5, wherein the variables included in the selected plural program cells are made sharable among the selected program cells when the selected program cells are combined by the combining process.

9. (Currently Amended) A compiler test system comprising:

test program generation means for generating a test program;

testing means for taking in the generated test program, compiling the test program by using a test target compiler and outputting an execution result;

expected value generating means for taking in the generated test program, compiling the test program by using a proven compiler and outputting an expected value; and

result comparison means for taking in the execution result and the expected value for comparison with each other and storing the comparison result in a storage unit as a test result,

wherein the test program generation means selects a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions, each of which may form an element of a test program for a compiler, and combines the selected sub-procedural descriptions to generate the test program;

wherein the sub-procedural description includes as a type of a basic sub-procedural description that lists executable statements in the order of execution, a control sub-procedural description that describes how to repeat a procedure, a control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function;

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result; and

wherein the statements for the stand-alone execution are made ineffective when compiling the test program.

10. (Cancelled)

11. (Currently Amended) A compiler test method comprising the steps of:

generating a test program;

taking in the generated test program, compiling the test program by using a test target compiler and outputting an execution result;

taking in the generated test program, compiling the test program by using a proven compiler and outputting an expected value;

taking in the execution result and the expected value for comparison with each other; and

storing the comparison result in a storage unit as a test result,

wherein the step of generating the test program comprises the sub-steps of:

selecting a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions, each of which may form an element of a test program for a compiler; and

combining the selected sub-procedural description to generate a test program;

wherein the sub-procedural description includes as a type of a basic sub-procedural description that lists executable statements in the order of execution, a control sub-procedural description that describes how to repeat a procedure, a

control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function;

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result; and

wherein the statements for the stand-alone execution are made ineffective when compiling the test program.

12. (Cancelled)

13. (Currently Amended) A program product for allowing a computer to execute a functional verification process for a compiler, said functional verification process comprising the steps of:

selecting a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions, each of which may form an element of a test program for a compiler;

combining the selected sub-procedural descriptions in accordance with the selected order to generate a test program;

taking in the generated test program, compiling the test program by using a test target compiler and outputting an execution result;

taking in the generated test program, compiling the test program by using a proven compiler and outputting an expected value;

taking in the execution result and the expected value for comparison with each other; and

storing the comparison result in a storage unit as a test result,

wherein the sub-procedural description includes as a type of a basic sub-procedural description that lists executable statements in the order of execution, a control sub-procedural description that describes how to repeat a procedure, a control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function;

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result; and

wherein the statements for the stand-alone execution are made ineffective when compiling the test program.

14. (New) An automatic compiler test program generation method, comprising the steps of:

selecting a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions, each of which may form an element of a compiler test program; and

combining the selected sub-procedural descriptions in accordance with the selected order to generate a compiler test program,

wherein the sub-procedural description includes as a type a basic sub-procedural description that lists executable statements in the order of execution, a

control sub-procedural description that describes how to repeat a procedure, a control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function;

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result; and

wherein the combining step combines the sub-procedural descriptions in such a manner as: if the selected sub-procedural description preceding the current sub-procedural description is the basic sub-procedural description or the function sub-procedural description, the current sub-procedural description is concatenated to the preceding sub-procedural description; and if the selected sub-procedural description preceding the current sub-procedural description is the control sub-procedural description, the current sub-procedural description is nested into the preceding sub-procedural description.

15. (New) An automatic compiler test program generation method according to claim 14, wherein each sub-procedural description is assigned a value which indicates the probability of the sub-procedural description being selected and the value is reflected in the random number-used selection of sub-procedural descriptions.

16. (New) An automatic compiler test program generation method according to claim 14, wherein variables included in the selected plural sub-procedural

descriptions are made shared by the selected sub-procedural descriptions when the selected sub-procedural descriptions are combined.

17. (New) An automatic compiler test program generation method, comprising the steps of:

selecting a plurality of program cells by a random number-used method from a set of program cells, each of which may form an element of a test program for a compiler after being compiled by the compiler; and

combining the selected program cells in accordance with the selected order to generate a test program,

wherein the program cell includes as a type a basic cell that lists executable statements in the order of execution, a control cell that describes how to repeat a procedure, a control cell that describes how to select a procedure, and a function cell that describes statements to call a function;

wherein the program cell includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result; and

wherein the combining step combines the program cells in such a manner as: if the selected program cell preceding the current program cell is the basic cell or the function cell, the current program cell is concatenated to the preceding program cell; and if the selected program cell preceding the current program cell is the control cell, the current program cell is nested into the preceding program cell.

18. (New) An automatic compiler test program generation method according to claim 17, wherein the variables included in the selected plural program cells are made sharable among the selected program cells when the selected program cells are combined by the combining process.

19. (New) A compiler test system comprising:

test program generation means for generating a test program;

testing means for taking in the generated test program, compiling the test program by using a test target compiler and outputting an execution result;

expected value generating means for taking in the generated test program, compiling the test program by using a proven compiler and outputting an expected value; and

result comparison means for taking in the execution result and the expected value for comparison with each other and storing the comparison result in a storage unit as a test result,

wherein the test program generation means selects a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions each of which may form an element of a test program for a compiler, and combines the selected sub-procedural descriptions to generate the test program;

wherein the sub-procedural description includes as a type a basic sub-procedural description that lists executable statements in the order of execution, a control sub-procedural description that describes how to repeat a procedure, a

control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function;

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result;

wherein the statements for the stand-alone execution are made ineffective when compiling the test program; and

wherein the combining step combines the sub-procedural descriptions in such a manner as: if the selected sub-procedural description preceding the current sub-procedural description is the basic sub-procedural description or the function sub-procedural description, the current sub-procedural description is concatenated to the preceding sub-procedural description; and if the selected sub-procedural description preceding the current sub-procedural description is the control sub-procedural description, the current sub-procedural description is nested into the preceding sub-procedural description.

20. (New) A compiler test method comprising the steps of:

generating a test program;

taking in the generated test program, compiling the test program by using a test target compiler and outputting an execution result;

taking in the generated test program, compiling the test program by using a proven compiler and outputting an expected value;

taking in the execution result and the expected value for comparison with each other; and

storing the comparison result in a storage unit as a test result,

wherein the step of generating the test program comprises the sub-steps of:

selecting a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions each of which may form an element of a test program for a compiler; and

combining the selected sub-procedural description to generate a test program;

wherein the sub-procedural description includes as a type a basic sub-procedural description that lists executable statements in the order of execution, a control sub-procedural description that describes how to repeat a procedure, a control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function;

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result;

wherein the statements for the stand-alone execution are made ineffective when compiling the test program; and

wherein the combining step combines the sub-procedural descriptions in such a manner as: if the selected sub-procedural description preceding the current sub-procedural description is the basic sub-procedural description or the function sub-procedural description, the current sub-procedural description is concatenated to the

preceding sub-procedural description; and if the selected sub-procedural description preceding the current sub-procedural description is the control sub-procedural description, the current sub-procedural description is nested into the preceding sub-procedural description.

21. (New) A program product for allowing a computer to execute a functional verification process for a compiler, said functional verification process comprising the steps of:

selecting a plurality of sub-procedural descriptions by a random number-used method from a set of sub-procedural descriptions, each of which may form an element of a test program for a compiler;

combining the selected sub-procedural descriptions in accordance with the selected order to generate a test program;

taking in the generated test program, compiling the test program by using a test target compiler and outputting an execution result;

taking in the generated test program, compiling the test program by using a proven compiler and outputting an expected value;

taking in the execution result and the expected value for comparison with each other; and

storing the comparison result in a storage unit as a test result,

wherein the sub-procedural description includes as a type a basic sub-procedural description that lists executable statements in the order of execution, a control sub-procedural description that describes how to repeat a procedure, a

control sub-procedural description that describes how to select a procedure, and a function sub-procedural description that describes statements to call a function;

wherein the sub-procedural description includes statements necessary for a stand-alone execution, a definition to define and initialize variables as well as to define constants and macros, and statements to display and store an execution result;

wherein the statements for the stand-alone execution are made ineffective when compiling the test program; and

wherein the combining step combines the sub-procedural descriptions in such a manner as: if the selected sub-procedural description preceding the current sub-procedural description is the basic sub-procedural description or the function sub-procedural description, the current sub-procedural description is concatenated to the preceding sub-procedural description; and if the selected sub-procedural description preceding the current sub-procedural description is the control sub-procedural description, the current sub-procedural description is nested into the preceding sub-procedural description.